FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Natural User Interfaces in Games for Mobile Devices

Adelino Manuel de Sousa Lobão

Master in Informatics and Computing Engineering

Supervisor: Rui Pedro Amaral Rodrigues
(Invited Auxiliar Professor at the Faculty of Engineering of the University of Porto)

Second Supervisor: António Fernando Vasconcelos Cunha Castro Coelho
(Auxiliar Professor at the Faculty of Engineering of the University of Porto)

13^{th} July, 2011

Natural User Interfaces in Games for Mobile Devices

Adelino Manuel de Sousa Lobão

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Jorge Manuel Gomes Barbosa (Auxiliar Professor at Faculty of Engineering of the University of Porto)

External: Pedro Miguel de Jesus Dias (Auxiliar Professor at University of Aveiro)

Supervisor: Rui Pedro Amaral Rodrigues (Invited Auxiliar Professor at Faculty of Engineering of the University of Porto)

13^{th} July, 2011

Abstract

Natural user interfaces are a relatively recent concept, they aim to eliminate the learning curve between the user and the interface, and translate natural body movements to actions.

A good example of the application of natural user interfaces is the mobile devices. The incorporation of touch, face and voice recognition, motion detectors in these devices, has unlocked new forms of interactions and improved the communication between the user and the interface.

One of the major drives of the incorporation of these interfaces in mobile devices was the iPhone and its multitouch interface. Never before has a mobile device had successfully implemented a non-tactile interface with so much success. Using the multitouch interface, it is possible to interact directly with the graphical elements displayed in the screen of the mobile device using multitouch gestures, improving the user experience and enhancing the interaction with the interfaces.

One of the fields that benefited with the introduction of multitouch technology in the mobile devices, was the mobile games field, previously the interactions were made mainly using keys. But using multitouch technology, enables the player to perform simple multitouch gestures in order to control the game, interacting more naturally with the game interface.

The work presented in this master thesis was divided in two different stages. The first stage was the development of a framework for the recognition of multitouch gestures. This stage also had the objective of exploring different types of multitouch gestures that could be used in the creation of new forms of interaction between the user and the mobile device.

The next stage was the development of a game for a mobile device that uses exclusively multitouch gestures as main interface for the communication with the players. The objective was to explore the incorporation of multitouch gestures has a main form of interaction. In order to evaluate this incorporation, it was performed a survey using real participants with the objective of measuring the usability of the game interface.

Resumo

Interface naturais são um conceito relativamente novo, têm o objectivo de eliminar a curva de aprendizagem entre o utilizador e a interface, e traduzir os gestos corporais em acções.

Um bom exemplo da aplicação das interfaces naturais são os dispositivos móveis. A incorporação de detectores de toque e movimento em dispositivos móveis permitiu o desenvolvimento de novas formas de interacção e melhorou o processo comunicação entre o utilizador e a interface.

Um dos maiores impulsionadores na integração deste tipo de interfaces em dispositivos móveis foi o iPhone e a sua interface multi-toque. Nunca um dispositivo móvel tinha obtido tanto sucesso na implementação de uma interface não táctil.

Através da interface multi-toque, é possível interagir directamente com os elementos que são mostrados no ecrã do dispositivo móvel utilizando gestos multi-toque, melhorando a experiência de utilização e interacção com as interfaces.

Uma das áreas que mais beneficiou com a introdução da tecnologia multi-toque nos dispositivos móveis, foram os jogos. Anteriormente as interacções eram realizadas através de teclas. Através da utilização da tecnologia multi-toque, permitiu que os jogadores executassem simples gestos multi-toque de forma a controlar o ambiente de jogo, interagindo mais naturalmente com a interface de jogo.

O trabalho descrito nesta tese encontra-se dividido em duas fases. A primeira fase é relativa ao desenvolvimento de uma *framework* para o reconhecimento de gestos multi-toque. Nesta fase o objectivo é explorar diferentes gestos multi-toque que possam ser utilizados na criação de novas formas de interacção entre o utilizador e o dispositivo móvel.

A fase seguinte é o desenvolvimento de um jogo para dispositivos móveis que utilize exclusivamente gestos multi-toque como interface de comunicação com os jogadores. O objectivo é explorar a integração de gestos multi-toque como forma principal de interacção. De forma a avaliar a incorporação dos gestos, será realizado um inquérito de forma a medir a usabilidade da interface de jogo desenvolvida.

Acknowledgements

I would like to thank you everyone who, somehow, helped me to accomplish this project.

My many thanks to the friends who accompanied me during these five years, without whom this would have been the most boring journey.

In particular, I would like to thank my supervisor, Prof. Rui Rodrigues, for the readiness he always showed to help me. And also, I would like to thank to Dr. Eduardo Carqueja the possibility to develop this thesis in a company environment and for the tools used in the thesis.

I would like to thank to my family, for giving me the possibility to study all these years, for the constant support during the project, and during my life.

Finally, I would like to thank you Inês Beirão, for your support in the bad moments and for always being at my side.

Thank you

Adelino Lobão

Contents

List of Figures

List of Tables

LIST OF TABLES

Abbreviations and Symbols

ANN	Artificial Neural Network
DTW	Dynamic Time Warping
GSS	Golden Section Search
GUI	Graphical User Interface
HCI	Human Computer Interaction
HMM	Hidden Markov Model
IDE	Integrated Development Environment
JSON	Javascript Object Notation
LED	Light-emitting Diode
NUI	Natural User Interface

Chapter 1

Introduction

Imagine a future for technology where humans and computers interact closely with the ability to communicate with gestures and touch, or having a computer able to recognize the voice and face of a person.

The use of this type of technology was just a dream a few years ago, but nowadays it is becoming reality. Touch, face and voice recognition, motion detectors are all part of a new system of information processing in full swing that is called natural user interface, or NUI.

When people interact with technology in this way, it becomes more human. Everyone can benefit from using natural user interfaces in ways that are more adaptable to the person, place, task, social context and mood. It promises a bright future in which an entire ecosystem of devices can coexist with humans in a way that makes life better for people.

One of the early technologies that applied the concept of natural user interfaces, was multitouch. In recent years, these systems have become the focus of a great deal of research and commercial activity. There have been a number of research efforts investigating compelling applications of these technologies, in examining alternative sensing methods, and in the design of interaction methods [WM10].

The evolution of mobile devices is one of the best examples of the integration of multitouch technology. These devices have become increasingly present in various contexts of personal and professional use. In this dissemination process, it became clear that the way that we interact with the device had to change, because using alphanumeric keyboards or single touch screens was limiting the type of interactions.

With the embedding of multitouch screens in mobile devices, the way in which a person interacts with these devices was totally changed. Using this technology, it is possible for a person to touch directly on the graphical elements displayed on the screen of the mobile device, improving the user experience.

One of the fields that benefited with the introduction of the multitouch technology on mobile devices, was the mobile games field. This technology unlocked new ways for the player to interact with game interface and made the interaction more natural [BIB+11].

The project herein described was conducted in a company environment. The company where the project was developed was the AppGeneration, is a software house that develops applications for mobile devices. And the challenge proposed was to explore new ways to interact with games that run on mobile devices, using the multitouch technology as the main form of interaction.

So the main objective of the project is to explore the inclusion of natural user interfaces in a mobile game environment by using multitouch technology as the main pillar of the project. To reach the objective is needed to build a framework for gesture recognition. And then, using the framework to explore the application of multitouch gesture in a mobile gaming environment.

1.1 Motivation

Mobile devices and the way that people interact with them are changing. Nowadays, the most advanced mobile devices have multitouch screens that enable the use of our fingers as input devices. The use of this technology enhances the user experience and improves the interaction between the user and the interface.

Despite of the advance of multitouch technology, most of the interactions implemented using it are very simple, through only one or two fingers in the interaction. An unexplored field in this area is the recognition of multitouch gestures and the exploration of the effect of the fusion of this type of gestures in a gaming environment.

Given that multitouch and gestures are natural user interfaces, this work will allow to study the inclusion of these type of interfaces in mobile games and understand how the players react to the use of such interfaces.

1.2 Thesis Objective

The main purpose of this thesis is to explore the incorporation of natural user interfaces in games for mobile devices. To achieve this objective, it is important to explore how a person interacts with a mobile device when using multitouch technology.

In order to support the research of the multitouch interactions between a person and a mobile device, it is necessary to develop a framework to recognize multitouch gestures. The algorithm of recognition used in the framework, needs to be light and flexible because it needs to run on a resource-constrained device.

Using the core of the multitouch framework, it will be developed a game that integrate a set of multitouch gestures. These gestures will be used in the communication process between the player and the game interface. The objective of the integration of the gestures is to enhance the user experience and make the interaction with the interface easier.

In the end, the usability of the interaction methods implemented in the game, will be evaluated. This evaluation will be a result of an analysis conducted with real users and with different backgrounds.

1.3 Document Structure

The remainder of this document is structured in the following manner. Chapter 2 presents the bibliographical revision of the literature and technology that supports this master thesis. The section 2.1 introduces the concept and characteristics of natural user interfaces. In section 2.2 is presented the concept behind the multitouch technology, the terms most used in this technology and a study about its advantages and limitations. Section 2.3 presents the state of art of mobile games that use multitouch as main form of interaction. And section 2.4 presents the most used techniques in the recognition of gestures and a critical analysis about the techniques presented.

Chapter 3 presents the framework developed for the gesture recognition. It is explained the algorithm developed and its limitations and improvements. In the end, the structure and data flow of the framework is presented in more detail

In chapter 4 is presented the game that was developed in order to explore the integration of multitouch gestures in a gaming environment. And in chapter 5 is presented the results of the research about the incorporation of natural user interfaces in game. Finally, this document ends with a conclusion and discussion about future work, on chapter 6.

Chapter 2

Bibliographical Revision

The current chapter introduces the bibliographical revision of the literature and technology that supports the work produced in this thesis. As previously mentioned, the objective of this project is to explore the integration of natural user interfaces in games in a mobile environment.

The section 2.1 introduces the concept of natural user interfaces and the section 2.2 presents one of the earliest technologies that supports the application of the principles of natural user interfaces, the multitouch technology.

In the section 2.3 is presented a study about the state of art of the games market on the mobile devices. The section 2.4 presents the most common algorithms used in gestures recognition and a critical analysis to these algorithms, with the objective to evaluate which is the best algorithm to use.

2.1 Natural User Interfaces

There is little doubt that there has been a surge of interest in the research of natural user interfaces such as multi-touch and gestural computing. That surge of research interest has been accompanied by the excitement about the commercial possibilities of these interfaces [SD10].

One of the best examples of an integration of these interfaces are the mobile devices. Nowadays it is normal to incorporate multi-touch screens in mobile devices. The use of this type of screens, allows the user to navigate through interfaces using multi-touch gestures, which enhances the user experience in these devices.

In this chapter the meaning of the term natural user interfaces is presented, together with a set of characteristics that defines it.

2.1.1 Definition

Natural user interface, or NUI, is a user interface that is easy to use and as seamless as possible. The perception of using these type of interfaces can be faded out by not appending any interaction devices to the user and by designing the communication process in a form that is really natural and understandable for the user [Rai99].

The natural user interfaces are an emerging paradigm in human-computer interaction which concentrates on human skills such as touch, vision, voice, motion and higher cognitive functions such as expression, perception and recall. It aims to take advantage of the power of a larger breadth of communication modalities, leveraging the skills that people earn through traditional physical interactions [Gro09].

Just as graphical user interface, or GUI, was a leap forward for computer users from command line interfaces. The natural user interfaces will become a common form of communication with computers.

The ability of humans and computers interacting in different and robust ways will release us from the current constraints of computers allowing complex interaction with digital objects in our physical world [Gro09].

2.1.2 Characteristics

This section presents the a of characteristics that defines a natural user interface, these characteristics are described by *Weiyuan Liu* in the article [Liu10].

- **User-centered**, is an interface where the needs, wants and limitations of the users are given extensive attention at each stage of the design process of the interface. It is a multi-stage problem, that not only requires the analysis and foresees how users are likely to use the interface, but also to test the validity of their assumptions with regards to user behavior in real world tests with real users.

- **Multi-channel**, is an interface that intends to make use of the sensory and motor channels to capture the complementary characteristics of the user's intention to enhance the naturalness of human-computer interaction. This process is reflected in an increased perception by the device about the user's intention, making the communication process easier.

- **Inexact**, is an interface that understands the user intentions. The user's actions or thoughts are not very accurate, so the interface should understand the user's request, and even correct some mistakes in the communication process, in order to understand what are the true intentions of the user.

- **High Bandwidth**, is an interface that is able to capture large quantities of information. Nowadays with the use of the keyboard as an input device, the input is captured

one after another hit, so, the input bandwidth is very low. But using a natural user interface in the interaction process, the user can use voice, image, etc. This means that the amount of information to capture is bigger, so, the interface needs a higher bandwidth.

- **Voice-based**, is an interface that uses voice-based interactions in the communication process with the user. The language is recognized as being the most natural convenient and efficient way of information-sharing. Using this channel, the communication experience is more natural and reliable.

- **Image-based**, is an interface that integrates image-based interactions in the communication process with the user. These interfaces are based in the human behavior of image analysis. First the image is processed in order to better understand its meaning and depending on the perception and analysis of the image, the interface transmits to the system the resulting action.

- **Behavior-based**, is an interface that uses behavior-based interactions in the communication process with the user. These interfaces recognize the human behavior through the positioning, tracking, movement and expression characteristics of human body parts, understanding which are the actions that the behavior describes.

2.2 Multitouch

One of the earliest technologies that supported the application of the principles of natural user interfaces, was multitouch.

Since the day that computers became a fundamental element in our society, the way that people interact with them has not changed, people continue to use keyboard, mouse or joystick as main input method. But to improve computers' usage, it is necessary to optimize the communication process between people and computers [HP95].

Direct manipulation is one of the most important concepts in human-computer interaction. The intention is to allow a user to directly manipulate objects presented to them rather than through an intermediary system [Wat93]. The more a user has to think about the interface, the more removed the user will feel from the task.

Multitouch technology refers to a touch sensing surface's ability to recognize the presence of two or more points of contact with the surface. Using this technology, the users can interact directly with the graphical elements that appear on their screen. One of the benefits of this technology is that interacting with an application that enables the direct touch on the graphical elements, is a more natural approach than working indirectly with a mouse or other pointing device [FWSB07].

The ability of using two hands improves the quality of interaction between the user and the interface, which means the appeal of direct manipulation with multitouch interfaces stems from the experience it offers. Another aspect that multitouch technologies provide is the ability of two or more people interacting simultaneously on the same screen, if the screen size allows.

At the moment there is no standard hardware to develop devices that integrate multitouch technology. It has been implemented in several different ways, depending on the size and type of interface. The most popular form are mobile devices (iPhone, iPod Touch, iPad), touch-tables (Microsoft Surface) and walls.

In this chapter is presented the basic vocabulary used in multitouch context, as well as the advantages and limitations of the use of this type of technology.

2.2.1 Terminology

This section contains the basic vocabulary used in multitouch context:

- **Tap**, happens when the user touches the screen with a single finger and then immediately lifts the finger off the screen without moving it around [ML08].
- **Touch**, refers to one finger being placed on the screen, dragging across the screen, or being lifted from the screen. The number of touches involved in a gesture is equal to the number of fingers on the screen at the same time [ML08].
- **Direct-touch** refers to the ability to touch a specific location on the display directly [KAD09]. The figure 2.1 exemplifies an direct-touch example.

Figure 2.1: Direct-touch.

- **Bimanual Interaction** refers to interaction done using both hands [KAD09]. The figure 2.2 exemplifies a bimanual interaction.
- **Multi-finger Interaction** refers to the use of multiple fingers on the interaction, but not necessarily with two hands [KAD09]. The figure 2.3 exemplifies a multi-finger interaction.
- **Gesture**, is any sequence of events that happens from the time the user touches the screen with one or more fingers until the user lifts all the fingers off the screen. No

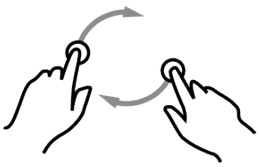

Figure 2.2: Bimanual Interaction.

Figure 2.3: Multi-finger Interaction.

matter how long it takes, as long as one or more fingers are still against the screen, the user is still within a gesture [ML08].

2.2.2 Advantages and Limitations

The integration of multitouch technology in the user interface has brought numerous advantages. It allows more parallel interactions, meaning that the interactions are done at the same time, decreasing the complexity of the user interface [Mos07]. And parallel interactions lead to better performance, because sub-task execution can overlap in time. Besides, each hand can remain in the proximity of the area of work which is responsible for, thereby reducing large-scale movements that may be required in interactions with one hand [DET90].

In the article [KAD09] a survey is described, with the objective of measuring the time that a user took to target a specific area on the display. The results show that the fastest touch-based selection was twice as fast as mouse-based selection. This reduction in time selection was due to the direct-touch nature of touch-based devices. So it is possible to conclude that using two fingers, one on each hand, and using any number of fingers does not increase the performance or efficiency.

Multi-touch devices detect multiple points of input, allowing the design of high degree of freedom interaction techniques that make use of people's manipulation skills. But, because of the physical limitations of the human hand, direct-touch interaction suffers from limited precision, occlusion issues and limitations on the size and proximity of the display [MH08].

It is possible to use two hands to increase interaction on multitouch surfaces. This type of interaction is especially well suited for high degree of freedom symmetric bimanual interactions such as shape deformations. One of the problems of using two-hands, is when two interactions are very near, it becomes difficult to distinguish which finger belongs to which hand [MH06]. And also, when several users are interacting in the same device, it is difficult for the device to distinguish the input of a specific user [WMW09].

Other problem of using touch devices, is that sometimes when interacting with objects, the user's hand and fingers obscure the object. Or when a finger is selecting something, the hand can cover some areas of the display, making it very difficult to interact with other areas of the screen as well as hiding possible crucial areas of interaction [MH08].

In order to prevent some limitations previously described, the article [KAD09] presents a set of design guidelines. These guidelines are applied where target selection is the primary task.

- A one finger direct-touch device, delivers a large performance gain over a mouse-based device. For a multi-target selection task, even devices that detect only one point of touch of contact can be effective.
- Support for detecting two fingers is enough to further improve performance of multi target selection.
- Reserve same-hand multi-finger usage for controlling multiple degrees of freedom or disambiguating gestures rather than for independent target selections.
- Uniformly scaling up interfaces, that were originally designed for desktop workstations for use with large display direct-touch devices, is a viable strategy as long as targets are at least the size of a fingertip.

2.3 Mobile Gaming

One of the fields that benefited with the introduction of multitouch technology in mobile devices, was the mobile games field. A game is an application, where a user interacts with an interface and feedback is generated based on the actions that the user takes.

Before the incorporation of the multitouch technology in mobile games, the interaction was made mainly using keys. Nowadays the mobile devices are equipped with multitouch screens, that allow the user to interact using touch.

The introduction of this interaction method changed the way that the user communicates with a mobile device. Using simple gestures, the user can control the game more naturally.

Therefore mobile game market has reached a very large scale. One of the major thrusters was Apple's iPhone which brought video games to vast new audiences who have

never been gamers before [Eco10]. The following games are a small subset of the market of games that use the touch as a main form of interaction.

- *Flight Control* is a strategy video game, where the user has to land the maximum number of planes, avoiding collisions between them. The user has to guide the aircrafts to their landing zones but avoid their collision. Using touch, the user selects the aircraft with the finger and then draws the trajectory of the aircraft to the landing zone. Figure 2.4 shows an example of a user landing an aircraft.

Figure 2.4: Gameplay of *Flight Control*.

- *Cut the Rope* is a physics-based puzzle video game. The objective of the game is to maneuver a piece of candy into the mouth of a cartoon monster, picking up three stars per level by touching them with the candy. The candy is hung by one or several ropes, using touch, the user cuts the ropes through the swipe of his finger in order to reach the objective of the game. Figure 2.5 shows an example of a user cutting a rope.

- *Gun Bros* is an action video game, where the user controls a soldier with the objective of defending the universe from alien attacks. The user controls the soldier through a virtual keypad, that allows the soldier to move and shoot against a horde of enemies. In the bottom of the figure 2.6 is possible to identify the virtual keypad used to control the soldier.

- *TwisTouch* is a multitouch puzzle video game. The game has fire balls in red, blue, green and yellow, that the user needs to drag into the matching goals. This means that the red fire ball, needs to be dragged into the red goal. But, while the user is dragging the fire ball, he has to touch in all the balls at the same time. It is not possible to just drag a fire ball at a time, or release the finger from the others, because the game will

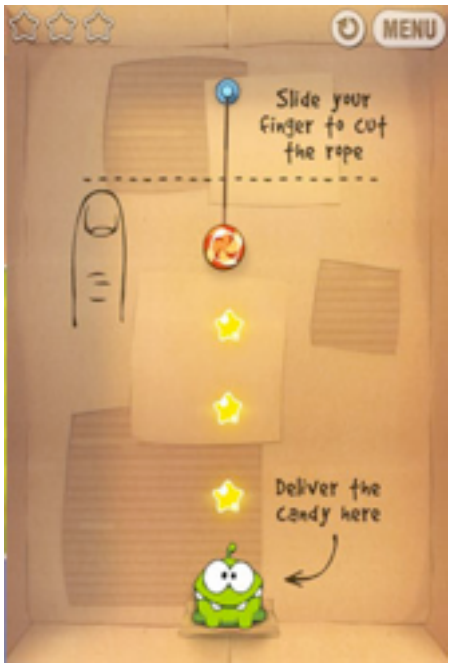

Figure 2.5: Gameplay of *Cut the Rope*.

Figure 2.6: Gameplay of *Gun Bros*.

start over. The gameplay is interesting because the user has to choose which hand or finger to use to accomplish the final goal of the game. Figure 2.7 shows an example of the gameplay, where the user is dragging two fire balls.

- *Multipong* is a multitouch and multiplayer version of the classic arcade game *Pong*. Since this is a multiplayer game, every player controls a bumper as a pinball bounces around. The goal is to bump the pinball past other player's bumpers without letting the pinball past our bumper. Using touch, the player uses the finger to control the

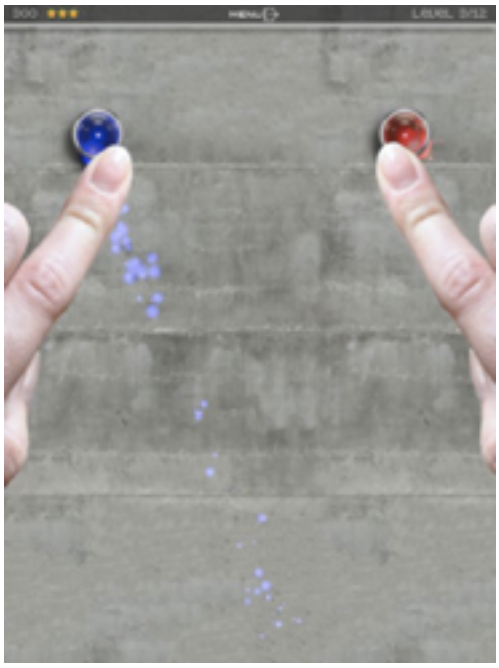

Figure 2.7: Gameplay of *TwistTouch*.

bumper, as it is shown in the figure 2.8.

Figure 2.8: Gameplay of *Multipong*.

- *Znipball* is a 4-finger action video game. The objective of the game is to shoot a puck into the opponent's goals. Using two fingers, the user catches the puck, stretches the rubber band, and then shoots the puck back into rival's goals. The players controls the bumper using the fingers like in the previous game.

- *Plants vs. Zombies* is a tower defense game. The objective is to stop a horde of zombies that want to attack the player's house. To stop the zombies, the player has to place different types of plants in their path to destroy them. Using touch, the player has to manage the plants and catch energy balls that enable to place more plants.

2.3.1 Conclusions

The games presented are a small sample of what exists in the mobile games market. After analyzing the games in a multitouch context, it is possible to identify that the interactions between the player and the interface are very simple. Normally the player has to drag objects across the screen or simply touch the object to catch it.

Despite of the player being able to use more than one finger to interact with the interface, normally each finger has an independent action.

It is possible to conclude that there are several unexplored areas in the use of multitouch in mobile games, especially:

- The use of more fingers to interact with the game.
- The usage of elaborated gestures in the game.

2.4 Methods for Multitouch Gesture Recognition

A gesture is a form of communication where the body executes specific movements in order to transmit a message to the receiver. Finger gestures are a specific type of gesture in which only fingers in contact with a surface are considered.

Recognition of gestures uses fingers as an input method, providing a meaning to these gestures through mathematical models. Being an expressive form of communication between humans, gestures allow users to understand their bodies as an input mechanism, without having to count on the restricted input capabilities of mobile devices [CPG07].

When implementing a gesture recognition system, attention must be given to pattern representation, feature extraction and selection, classifier design and learning, training and testing, and performance evaluation [JDM00]. Some of the best approaches to pattern recognition are:

- Template matching
- Statistical classification
- Syntactic or structural matching

This section provides a description of the most tried and tested gesture recognition techniques and methodologies,along with the various strengths and weaknesses of these techniques. In the end is presented, a critical analysis of each technique based on the performance of the gesture recognition.

2.4.1 Hidden Markov Model

A hidden Markov model, or HMM, is based on statistical classification technique where the system is a Markov process with an unobserved state. In probability theory, the Markov process is a time-varying random phenomenon for which a specific property (the Markov property) holds [RJ86].

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a HMM, the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states [RJ86].

A HMM can be considered a generalization of a mixture model where the hidden variables, which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other.

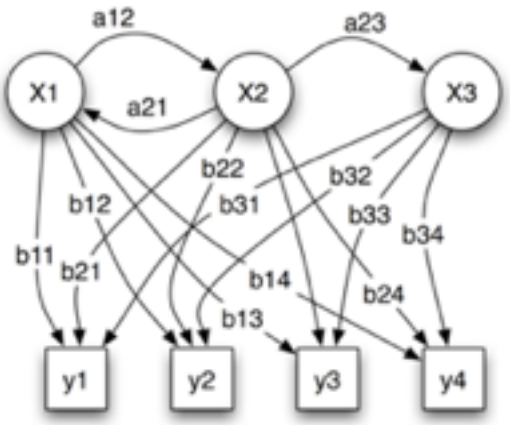

Figure 2.9: Structure of a HMM.

Figure 2.9 shows an example of the probabilistic parameters of a HMM. The elements *x's* are the states of the model, the *y's* are the possible observations, the *a's* are the state transition probabilities and the *b's* are the output probabilities.

HMMs use only positive data and they are scalable, meaning that new gestures may be added without affecting already learnt HMMs. One of the problems using hidden Markov models in gestures recognition is that HMMs requires a large amount of data to train.

2.4.2 Artificial Neural Networks

An artificial neural network, or ANN, is a computational model that is inspired by the structure and functional aspects of biological neural networks. It is based on the statistical

classification technique and is a network of weighted, directed graphs where nodes are artificial neurons, and the directed edges are connections between them [NK08].

A neural network processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools, usually used to model complex relationships between inputs and outputs or to find patterns in data.

The most common ANN structure is the feedforward multi-layer perceptron. Feedforward means that the signals only travel one way through the net. Neurons are arranged in layers, with the outputs of each neuron in the same layer being connected to the inputs of the neurons in the layer above [Hea08].

The ANNs consist of three layers, namely the input layer, hidden layer and the output layer. If the ANN is used into a gesture recognition context, once a gesture has been performed, its features are broken down into parts before being passed to different neurons of the input layer. The data is then passed to the hidden layer, where decisions are made (based on weightings) in order to classify the gesture; there may be more than one hidden layer. Finally, the output layer neurons are assigned a value. Each output layer neuron represents a particular class of gesture, and the record is assigned to whichever class's neuron has the highest value [Day90].

During training, the gesture class for each neuron in the output layer is known, and the nodes can be assigned the correct value (0.9 for the node corresponding the correct class, and 0.1 for the other). Consequently, it is possible to compare the ANNs calculated values for these node to the correct values they have been assigned, and hence calculate an error term. These error terms can then be used to adjust the weightings in the hidden layers, and thereby training the network to better calculate the correct values [Day90].

Although powerful, using and implementing ANNs requires large training times and a large sets of gestures. However, once the training has been completed, the classification of the gesture is very fast [DO96].

2.4.3 Dinamic Time Warping

Dynamic time warping, or DTW, introduced by *Sakoe and Chiba*, is an algorithm that compares two different sequences that may possibly vary in time [SC78]. It is based on the template matching technique.

For example, if two video clips of different people walking a particular path were compared, the DTW algorithm would detect the similarities in the walking pattern, despite walking speed differences, accelerations or decelerations during the course of one observation.

In general, DTW is a method that allows to find an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are *warped* non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension [SC78].

The algorithm begins with a set of template streams, describing each gesture available in the system database. Taking an unlabeled input stream of templates, the minimum distance between the input, and each template stream is calculated, and the input stream is classified according to the template stream it matches most closely. The warping comes in with an algorithm that is used to search the space of mappings from the time sequence of the input stream to that of the template streams, so that the total distance is minimized [Kad02].

The DTW has several weaknesses: it has a high computation time and the storing of many templates for each gesture results in costly space usage on a resource-constrained devices, as the mobile devices.

2.4.4 $1 Recognizer

The $1 Recognizer is a easy, cheap, and usable algorithm. It is based on the template matching technique and was developed to facilitate the incorporation of touch gesture into user interface prototypes.

The recognizer involves only basic geometry and trigonometry. It supports configurable rotation, scale, and position invariance, does not require feature selection or training examples, is resilient to variations in input sampling, and supports high recognition rates, even after only one representative example [WWL07].

The main goals of this algorithm are to:

- Be resilient to variations in sampling due to movement speed or sensing.
- Support optional and configurable rotation, scale, and position invariance.
- Require no advanced mathematical techniques.
- Be easily written in few lines of code.
- Be fast enough for interactive purposes.
- Allow the developers and end-users to *teach* new gestures, to the algorithm, with only one example.
- Provide recognition rates that are competitive with more complex algorithms used in HCI to recognize different types of gestures.

2.4.5 Critical Analysis

This section presents a critical analysis of the different techniques previously described, this analysis is based on the results achieved by [Nie08] and [WWL07].

- **ANNs, HMMs and DTW**

In the analysis presented in [Nie08] the ANNs, HMMs, and DTW algorithms were used in the recognition process of a set of gestures. The algorithms were implemented on a mobile phone, and measured in performance according the recognition speed, accuracy and time needed to train.

The HMMs were implemented with both 4 states, and 8 states, but it was found that 4 states were not able to correctly classify enough samples to accurately compare to other algorithms. Therefore, 4 states were not considered. For HMMs with 8 states, 77 of 80 samples were correctly classified after training. DTW also identified 77 samples correct, and ANN classified only 72 samples correctly [Nie08].

The results obtained in [Nie08] are summarized in the table 2.1.

Algorithm	Recognition Speed	Accuracy	Training Time
HMMs	10.5ms	95.25%	Long
ANNs	23ms	90%	Medium
DTW	8ms	95.25%	No Training

Table 2.1: Results of critical analysis described in [Nie08]

- **$1 Recognizer and DTW**

In the analysis presented in [WWL07] the $1 Recognizer and DTW algorithms were used to recognize 16 different gestures. The algorithms were implemented on a mobile phone, and measured in performance according to the accuracy, error and number of templates used in the recognition.

The $1 Recognizer performs very well for user interface gestures, recognizing them with more than 99% accuracy overall. DTW performed almost identically, but with much longer processing times. Both algorithms did well even with only 1 template, performing above 97% accuracy. With only 3 loaded templates, both algorithms functioned at about 99.5% of accuracy [WWL07].

- **Conclusions**

By analyzing the results in table 2.1, it is possible to see that the DTW algorithm is the most efficient algorithm in both recognition speed and training time, among HMMs and ANNs. And from the analysis of the results presented in [WWL07], the $1 Recognizer

and the DTW have almost the same accuracy and use the same number of templates in the recognition process.

So it is possible to conclude, that the $1 Recognizer and the DTW are the algorithms that have the best results in the recognition accuracy, although the DTW has longer processing times. Since the algorithm is going to be implemented on a mobile gaming environment, a lower processing time is one of the most important characteristics that the algorithm needs to satisfy because it has a direct impact in the playability of the game.

Therefore, the algorithm that offers the best balance between accuracy in the recognition and processing times for the needs of the current project is the $1 Recognizer.

2.5 Summary

Natural user interfaces are an emergent paradigm of human-computer interactions. In this chapter, this new concept was introduced, presenting its definition and attributes. Then it was presented one of the earliest technologies that applied the concept of natural user interface, the multitouch.

After the definition of these concepts, it was presented a small subset of the market of games, on mobile devices, that uses the multitouch as a main form of interaction, with the objective of evaluating the state of art of it. It was concluded that most of the interactions were very simple, and there were several unexplored areas, especially in the usage of multitouch gestures that enables the player to use the hand to perform gestures that enhance the communication with the game interface.

Based on the conclusions of this study, it was presented the most common algorithms used in the recognition of gestures. And to evaluate which was the best algorithm to use in the recognition process of multitouch gestures on a mobile device, it was presented a critical analysis based on the work developed in [Nie08] and [WWL07]. After analyzing with more detail these two studies it was possible to conclude that the best algorithm to use is the $1 Recognizer.

Chapter 3

Gesture Recognition

This chapter presents the solution that was created to recognize different types of multi-touch gestures. The solution is applied on a mobile context, so one of the most important things to have in mind is that it will run on a device with limited resources.

The solution that was developed, is an extension of the work presented by *Woobrock, Jacob*; *Wilson, Andrew* and *Li, Yang* described in [WWL07], this work describes a basic gesture recognizer that involves only geometry and trigonometry in the recognition process. The section 3.1 presents in more detail this solution and its limitations and improvements.

A framework was developed in order to evaluate the performance of the recognition process of the solution. This framework provides a set of tools to:

- Recognize which was the gesture drawn in the mobile device.
- Save new gestures to be used in further recognition.
- Manage all the gestures in the framework.

The framework is useful in the process of prototyping the gestures, because with the tools that were previously presented, the user can improve the success rate of the recognition process of any gesture that needs to be used.

3.1 Recognition

As previously introduced, the solution is based in the work described in [WWL07], whose algorithm is named *$1 Recognizer*. This algorithm is easy, cheap, and usable almost everywhere. Despite its simplicity, it offers 99% accuracy with only a few loaded templates. It requires no complex mathematical procedures but is able to compete with approaches that use dynamic programming and statistical classification [WWL07].

Although it has advantages, the *$1 Recognizer* cannot be used in the recognition of multitouch gestures, because it was developed only to recognize gestures with one touch.

In order to support multitouch gestures, a novel solution was developed, based on the *$1 Recognizer*. The main changes were focused in the preprocessing of the raw input data that represent the gesture, this process is described with more detail in the section 3.1.2.

Two of the most important features that the solution needs to accomplish are:

- Flexibility, because the algorithm needs to support a variety of different types of gestures and offer a easy way to add new gestures
- Speed of recognition, because it has a direct impact in the playability of a game, which is the area where the algorithm will be used

The following sections will present in more detail the solution. Section 3.1.1 describes the technique on which the solution was based. Section 3.1.2 presents the preprocessing that is applied to the raw input data, in order to prepare the gesture to the recognition process. Section 3.1.3 describes the matching process between the gesture and templates. Section 3.1.4 presents the limitations of the solution and section 3.1.5 describes some proposed improvements that would optimize the success rate of the recognition process and also correct some of the limitations presented in section 3.1.4.

3.1.1 Overview

Although there are several techniques used in gesture recognition, the most important are described in the chapter 2. The solution developed in this master thesis is based in the *Template Matching* technique.

Template Matching is one of the simplest techniques for gesture recognition. Essentially there is no representation stage, the *raw* sensor data is used as an input to the decider which is based on a function that measures the similarity between templates and the input. The input is classified as being a member of the same class as the templates to which it is most similar. In general, there is a similarity threshold, below which the input is rejected as belonging to none of the possible classes (or too far from the nearest template). In fact the similarity function is most often the euclidean distance between the set of observed sensor values and a template [Wat93].

Despite of using as reference the *Template Matching* technique, the solution presented does not use *raw* data for the matching. Whenever the solution receives a new gesture to recognize or store as template, it undergoes a procedure that normalizes the *raw* input data, that represents the gesture, so that it could be compared with other gestures.

3.1.2 Preprocessing

The preprocessing of the *raw* input data is one of the most important stages of the solution. In this stage, there are gestures meant to serve as a template or as a candidate attempting to

be recognized, but either way, they are initially treated as the same. The objective of this treatment is to normalize the *raw* input data, so that it could be in the same state as the data stored in the solution.

Before describing the process, it is important to explain some key concepts:

- **Point** is a coordinate that represents a position whereby the finger passed.
- **Trace** is a route of a finger. A gesture has the same number of traces as the number of fingers that belongs to the gesture.

The following sections describes the treatment that is applied to the *raw* input data.

3.1.2.1 1^{st} Step - Order traces

The solution receives the *raw* input data, that was determined by sensing hardware and input software. Since there are gestures with more than one finger, it is imperative to establish a natural order for the different traces that integrate the gesture. If the algorithm does not order the traces, it will not make the match between them.

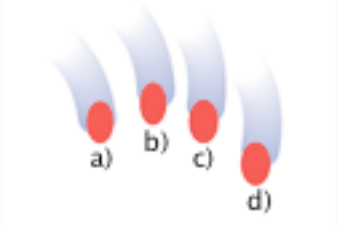

Figure 3.1: Gesture with four fingers.

The figure 3.1 represents a random gesture with four fingers. The gesture could have various representations of the *raw* input data, depending on when the user touched the screen with each finger, for example:

- First Scenario
  - 1^{st} finger: a)
  - 2^{nd} finger: b)
  - 3^{rd} finger: c)
  - 4^{th} finger: d)
- Second Scenario
  - 1^{st} finger: d)
  - 2^{nd} finger: c)
  - 3^{rd} finger: b)

– 4^{th} finger: a)

The two scenarios above show different representations of the gesture in the figure 3.1. Supposing that the first scenario represents a gesture to be recognized and the second scenario represents a template. The success rate of the matching between these two scenarios would have an error because the comparison of each trace was not done with the correct trace. To prevent this type of error, it is necessary to order the traces of the gesture following a standard procedure.

The first point of each trace is used to order the traces and then, with all of these points collected, a centroid is calculated. As it is showed in the figure 3.2, the red dots are the first points of each trace and the blue dot represents the centroid.

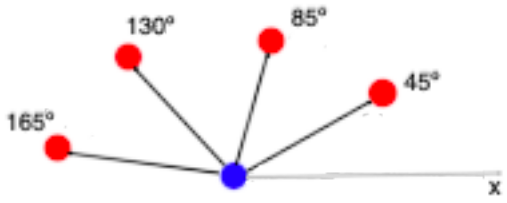

Figure 3.2: Procedure for ordering the traces of a gesture.

The next step is to calculate the angle between the centroid and the first point of each trace, in figure 3.2 each red dot has an associated angle. To complete the procedure, the traces are ordered according to the angle made between the first point of each trace and the centroid.

3.1.2.2 2^{nd} Step - Resample the point path

The resampling of each trace is necessary because the movement speed has a clear effect on the number of input points in a gesture [WWL07].

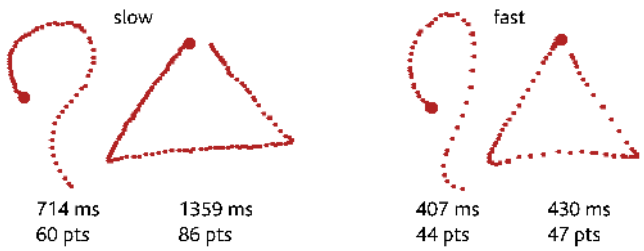

Figure 3.3: A slow and fast question mark and triangle. Note the considerable time differences and resulting number of points.

In the figure 3.3, it is possible to see the effect of the speed on each gesture. The gestures that were executed more slowly have a higher number of points relatively to gestures that were executed faster.

To make each trace of the gesture directly comparable with each other, even at different movement speeds, it is necessary to resample each trace such that the path defined by their original points is defined by N equidistantly space points, as the example of the figure 3.4.

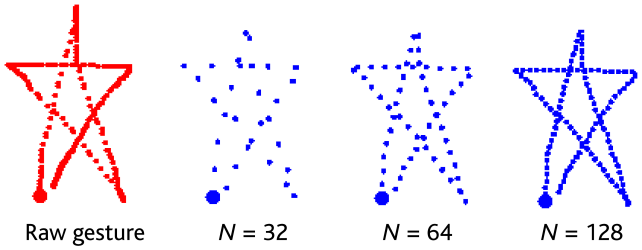

Figure 3.4: A star gesture resampled to N=32, 64 and 128 points.

The problem of using N equidistantly space points, is that if the value of N is too low, it will result in a loss of precision, while using a value of N that is too high adds more time to path comparison [WWL07]. This limitation of the algorithm is referred in the section 3.1.4.

3.1.2.3 3^{rd} Step - Rotate once based on the *Indicative Angle*

The algorithm has to compare two paths of ordered points, the gesture and template. There is no closed-form solution for determining the angle which one set of points should be rotated to best align with the other [KS04].

Instead of using brute force to calculate which is the better angle to compare the gesture with the template, it is possible to do a rotation that makes finding the optimal angle of comparison much faster [WWL07].

The algorithm needs to find the gesture's *indicative angle*, which is defined by the angle formed between the centroid of the gesture and the gesture's first point. Then rotate the gesture so that this angle is at 0^o. This process is showed in the figure 3.5.

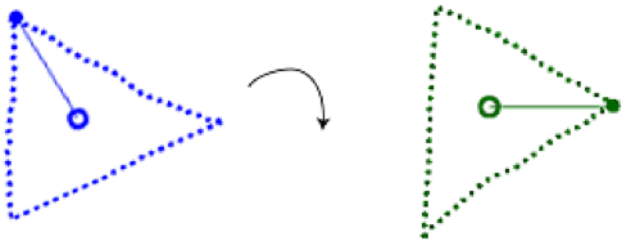

Figure 3.5: Rotating a triangle so that its *indicative angle* is at 0^o.

3.1.2.4 4^{th}Step - Scale and Translate

After rotation, the gesture is scaled to a *reference square*. By scaling it to a square, the algorithm is scaling non-uniformly. This allows to rotate the candidate around its centroid and safely assume that changes in pairwise point-distance between the gesture and templates are due only to rotation, not to aspect ratio.

After scaling, the gesture is translated to a reference point which, to simplify, is the origin (0,0).

3.1.3 Matching Process

When the algorithm finishes processing the *raw* input points, the data that represents the gesture is in the same condition as the templates. To finish the recognition process, the algorithm needs to match the candidate gesture with all the templates, in order to find the template that is more similar.

Using equation 3.1, the candidate gesture C is compared to each stored template T_i to find the average distance d_i between the two gestures.

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N} \tag{3.1}$$

The result of equation 3.1 defines the distance between C and T_i. The template T_i with the smallest distance to C will be the result of the recognition.

When the algorithm compares the gesture C with the templates T_i, the result of each comparison must use the best angular alignment between C and T_i. In section 3.1.2.3, rotating C and T_i once using their indicative angles only approximated their best angular alignment. However, the gesture C may need to be rotated to find the smallest distance to the templates T_i [WWL07].

There is not a simple way to rotate C into T_i, the algorithm must fine-tune C's angle so that C's distance to T_i's is minimized. The brute force scheme could rotate C by 1^o for all 360^o and take the best result. Although this method guaranteed to find the best angle, it is unnecessary slow and could be a problem in processor-intensive applications.

To find the best angle that minimizes the distance between the gesture C and the templates T_i, the technique *Golden Section Search (GSS)*, described in [NPM+03], was used.

The GSS is a technique for finding the extremum (minimum or maximum) of a unimodal function by successively narrowing the range of values inside which the extremum is known to exist. The technique derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio ($\varphi = 0.5(-1+\sqrt{5})$. The algorithm is closely related to a Fibonacci search and to a binary search.

Based on the conclusions presented in the work [WWL07], no matches were found between gestures and templates, beyond $\pm 45^{o}$ from the indicative angle. So in order to optimize the algorithm, GSS uses $\pm 45^{o}$ as a maximum search angle and 2^{o} threshold.

3.1.4 Limitations

The solution presented is a simple algorithm used in the recognition of multi-touch gestures, but it has some limitations such as:

- It is not able to distinguish which hand made the gesture. For example, in figure 3.2, the gesture is what matters regardless of which hand made it.
- It does not distinguish if there is more than a hand involved in the gesture. As presented in section 3.1.2 the algorithm only performs basic geometry and trigonometry operations. The transformations that are applied to the *raw* input data are not enough for determining if the gesture was executed with more than one hand.
- Since the algorithm does not consider time, gestures cannot be differentiated based on the speed of execution of the gesture.
- Section 3.1.2.1, describes the process of ordering the traces of the gesture. Despite its simplicity, this process introduces limitations that will have impact in the success rate of the algorithm.

  To order the traces, the algorithm uses the first points of each trace to calculate the centroid between them. Then it calculates the angle between the first point of each trace and the centroid, as the figure 3.2 shows. To finish, the traces are ordered based on the angle of their first points.

  The limitation of this process is when the algorithm is calculating the angle between the first points and the centroid. The figure 3.6 shows the same gesture as in the figure 3.2 but with a small variation in the first position of each finger.

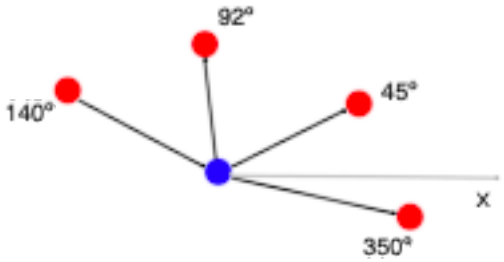

Figure 3.6: Procedure of ordering the traces of a gesture.

The gestures are the same but the order of the traces are different, the point with 350^{o} in the figure 3.6 has 45^{o} in the figure 3.2. This means that the trace that contains the

point with 45º will be the first trace of the gesture and the trace that contains the point with 350º will be the last trace of the second gesture. These errors in the order of the traces have a direct impact in the success rate of the recognition of the gesture.

To correct this limitation, it is necessary to define more templates in specific positions. But the number of templates that are necessary will depend on the number of fingers of the gesture.

- The number of sample points that is used to resample the traces has a direct impact on the success rate of the recognition process. Since the sample points are equally spaced between them:
  - If the number of sample points is too low, it would result in a loss of precision, while using a number of sample points too high adds time to path comparison.
  - If the traces, of the gesture, are long and irregular (have curves), the distribution of the sample points could erase vital information to identify the gesture.

3.1.5 Possible Improvements

Despite the good success rate that the solution obtains in the recognition process, there are improvements that could optimize it.

The improvements described in this section were not implemented, these are suggested improvements that will improve the success rate of the recognition process.

3.1.5.1 Sampling process

In section 3.1.4, two limitations that were referred, were related with the sampling of the gesture.

To improve the sampling process, instead of distributing the sample points equally spaced between them, one solution could be to use the concept of the *Ramer-Douglas-Peucker* algorithm presented in the article [DP73]. It is an algorithm aimed at reducing the number of points in a curve that is approximated by a series of points.

The purpose of the algorithm is, given a curve composed of line segments, to find a similar curve with fewer points. The algorithm defines *dissimilar* based on the maximum distance between the original curve and the simplified curve. The simplified curve consists of a subset of points that defined the original curve [DP73].

The figure 3.7 shows an example of a gesture that was treated by the *Ramer-Douglas-Peucker* algorithm, the gesture a) is the *raw* input data and the gesture b) is the result of the processing of the algorithm.

This improvement would have a direct impact in the success rate of the recognition process, specially in gestures with:

Figure 3.7: Smoothing a piecewise linear curve with the Douglas–Peucker algorithm.

- Low number of points, because it maximizes the distribution of the points.
- High variation between points (curved gesture).

3.1.5.2 Normalization Algorithm variant - Protractor

The Protractor is a gesture recognizer developed by *Yang Li* described in [Li10] and it is very similar to the *$1 Recognizer*. It uses a nearest-neighbor approach, which recognizes an unknown gesture based on its similarity to each of the known gestures, e.g., training samples or examples given by the user. In particular, it employs a method to measure the similarity between gestures, by calculating a minimum angular distance between them with a closed-form solution [Li10].

Protractor's preprocessing is similar to the section 3.1.2, but with several key differences in handling orientation sensitivity and scaling. Protractor gives the developer an option to specify whether it should work in a *orientation invariant* or *sensitive way*. In the *orientation invariant*, the gesture is rotated around its centroid by its *indicative angle*, as in the section 3.1.2.3. In the *orientation sensitive*, the Protractor aligns the indicative orientation of a gesture with one of eight base orientations that requires the least rotation, see figure 3.8.

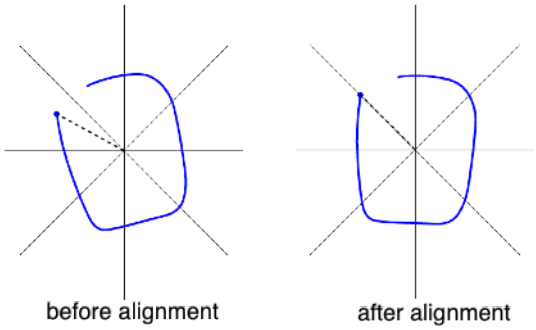

Figure 3.8: Aligning the indicative orientation of a gesture with the closest direction of the eight major directions.

The Protractor does not rescale the resampled points to fit a square, preserving the aspect ratio of a gesture and also making it possible to recognize narrow gestures such as horizontal and vertical lines. Rescaling these narrow gestures would seriously distort them and amplify the noise in trajectories.

But the most significant difference that Protractor presents is the employment of a closed-form solution to find a rotation that leads to the minimum angular distance. For each pairwise comparison between a gesture template t and the unknown gesture g, Protractor uses the inverse cosine distance between their vector points, v_t and v_g, as the similarity score S of t to g, in the equation 3.2.

$$S(t,g) = \frac{1}{arcos\frac{v_t.v_g}{|v_t||v_g|}} \tag{3.2}$$

The cosine distance essentially finds the angle between two vectors in a n-dimensional space. As a result, the gesture size, reflected in the magnitude of the vector, becomes irrelevant to the distance. So Protractor is inherently scale invariant. The cosine distance of two vectors is represented by the dot product of the two vectors (equation 3.3) divided by the multiplication of their magnitudes (equation 3.4).

$$v_t.v_g = \sum_{i=1}^{n}(x_{ti}x_{gi} + y_{ti}y_{gi}) \tag{3.3}$$

$$|v_t|\,|v_g| = \sqrt{\sum_{i=1}^{n}(x_{ti}^2 + y_{ti}^2)}\sqrt{\sum_{i=1}^{n}(x_{gi}^2 + y_{gi}^2)} \tag{3.4}$$

However, it can be suboptimal to evaluate the similarity of two gestures by just looking at the angular distance calculated by the equation 3.2. So Protractor rotates a template by an extra amount so that it results in a minimum angular distance with the unknown gesture and better reflects their similarity. The optimal amount of θ that the template t needs to be rotated is given by the equation 3.5.

$$\theta_{optimal} = \arctan\frac{\sum_{i=1}^{n}(x_{ti}y_{gi} - y_{ti}x_{gi})}{\sum_{i=1}^{n}(x_{ti}x_{gi} + y_{ti}y_{gi})} \tag{3.5}$$

With the $\theta_{optimal}$ calculated, it will be easy to acquire the maximum similarity (the inverse minimum cosine distance) between the gesture and the template.

3.2 Framework

The previous solution was embedded on a framework used in the prototyping process of different multitouch gestures. Through the framework, the developers can define multiple template gestures and use them further in recognition processes. This process is useful, if the developers want to integrate the prototyped gestures in future applications. The framework structure is divided in four components, showed in figure 3.9.

- The *Capture* component is responsible for capturing the gesture performed by the user.

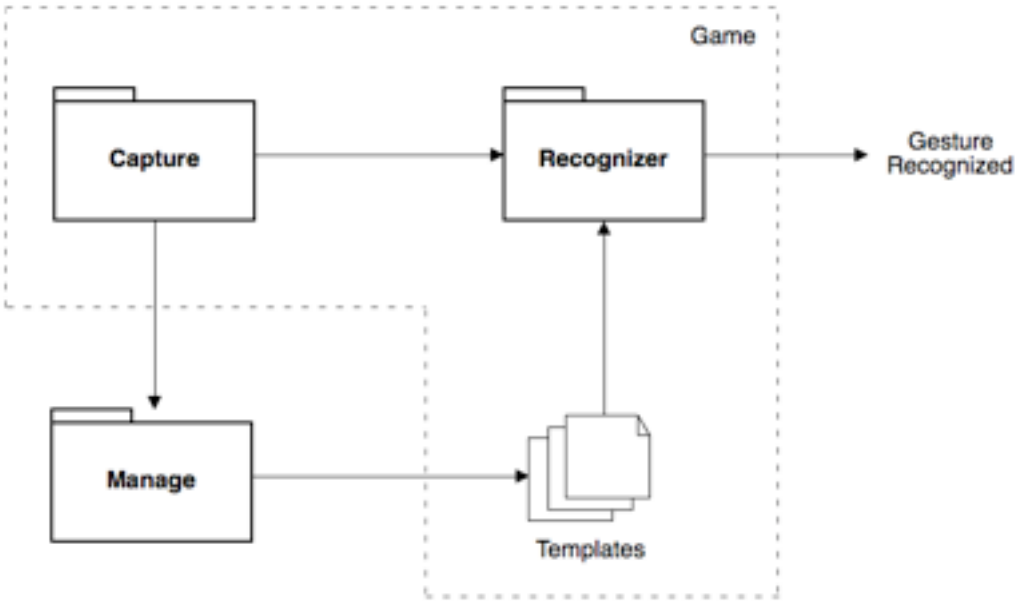

Figure 3.9: Structure of the framework.

- The *Recognizer* component is responsible for the recognition process of the gesture captured.
- The *Manage* component is responsible for organizing all the received gestures.
- The *Templates* component is a set of templates that are used in the recognition process.

In the prototyping process, the developer can save multiple gestures, that will be stored as templates. For each saved template, the developer has to introduce the name which identifies the template in the recognition process. To improve the success rate of the recognition, it is possible to save multiple gestures from the same template.

When the developer performs a gesture that he wants to be recognized, the recognizer compares it with all the templates in the framework, and then the algorithm returns the name of the template that has the best success rate.

The section 3.2.1 describes the different states of the data inside the framework. The section 3.2.2 describes the structure of the framework and how to integrate it into other applications. And the section 3.2.3 presents the development environment of the framework.

3.2.1 Data Flow

The data that represents the gestures undergoes multiple states. This section describes these different stages from the moment that the user touches the screen of the device until

the framework displays the name and success rate of the recognition process. The data could be represented as:

- *Raw* input points. This state represents a gesture without any preprocessing treatment, which means the data is in a raw state.
- Processed data. This state is when the data already suffered the preprocessing treatment and is ready to be used in the recognition process. The data could have two different representations:
  1. A gesture, that needs to be recognized.
  2. A template, that will be saved to further recognition.

The gesture starts when the screen is touched with a finger. The framework starts tracking the fingers of the gesture and building a structure that contains all the *raw* input points of the gesture.

When the gesture is finished, it undergoes a preprocessing treatment that is described in the section 3.1.2. The treatment transforms the *raw* input points into a readable data that the algorithm understands.

The meaning of this data depends on its functionality. If it is to be recognized, it means that the data is a gesture. If the data is to be saved for further recognition processes, it means that the data is a template.

3.2.2 Structure framework

Each component has a specific task, the *Capture* is responsible for capturing the gestures drawn in the screen. If the objective is to recognize the gesture performed, the information capture goes to the *Recognizer*. If the objective is to saved the gesture performed for further recognition processes, the information captured goes to the *Manage*.

Both the components *Capture* and *Recognizer* use the *Templates* to save and load, the system templates.

As previously mentioned, the framework could be integrated into other applications. In the chapter 4 is described a game that uses the same components of the framework to understand the meaning of the gestures that the players perform.

It is possible to identify in the figure 3.9 the components that were used in the game, such as: *Capture*, *Recognizer* and *Templates*. The component *Manage* was not included because the game already had defined the templates to use in the recognition process.

3.2.3 Development Environment

The framework was developed into two stages. In the first stage, a prototype of the framework was developed, using the library for multitouch development PyMT and the

programming language Python. The objective of this prototype was to essentially test the algorithm.

The next stage, was to implement the framework on an iPad, a mobile device of Apple under the iOS operative system. The programming language used was Objective-C.

Both stages were developed on a MacBook Pro 1,2 running Mac OS X 10.6.7 (Snow Leopard).

Chapter 4

Game

To explore the effect of the inclusion of natural user interfaces in a gaming environment an adventure game was developed. The developed game uses the framework for gesture recognition, described in chapter 3, for the communication process between the player and the game.

The player interacts with the game using predefined gestures to control its character. Each gesture executes a specific movement, for example:

- Moving the character through the level.
- Performing an attack to destroy an enemy.
- Interacting with an item in the level.

An important consideration is that the game itself had to be of a relatively limited scope, given the time constraints for development. However, the game should still be sufficiently addictive and extremely easy to play, as the main purpose is to measure the usability and playability of the game using multitouch gestures as the main forms of interaction.

The following sections presents with more detail the game. A small overview of the story and genre of the game is presented first, followed by the aspects related with the design and implementation.

4.1 Game Overview

The video game developed is a two-dimension adventure game. In this genre the player assumes the role of protagonist in an interactive story driven by exploration and puzzle-solving. Nearly all adventure games are designed for a single player, as the emphasis on the story and character makes multi-player design difficult [Ada09].

The game puts the player in the role of a driller that needs to catch all the gems in each level. But to reach the gems, the player needs to overcome several obstacles, because the

path is full of enemies and puzzles. The player has to plan cautiously which movements to perform, or else he may get stuck in a position where he will never be able to reach the gems, and having to restart the level.

Normally the interactions between the player and the interface, in an adventure game, are done using a game controller. But since the main focus of this game was to explore the integration of the natural user interfaces, the most important part on the development of the game was to understand how to incorporate the multitouch gestures in the gameplay of the game.

The gestures were chosen based on the perception of the different actions that the player needs to perform. For each action, different gestures were tested that best represented the meaning of the action and then the best one was selected. Each gesture was carefully tested to optimize its success rate.

4.2 Game Design

The current section presents the game design, an example of the game interface is presented in figure 4.1.

The section 4.2.1 presents the description of each multitouch gesture that integrates the game. It is using these gestures that the player interacts with the interface game.

The section 4.2.2 presents the description of the elements used to build the game level . An element can be a piece of the scenario, an item of the game, an attack that the player can perform or even a character of the game.

The section 4.2.3 presents the game interface and also describes the most common interactions between the player and the interface.

4.2.1 Game Gestures

The multitouch gestures that integrate the game were chosen based on the perception of the different actions that the player needs to perform to reach the final objective. The shape of how each gesture is drawn was conceived taking into account the way the action is performed in real life.

The multitouch gestures are grouped into three different types, each group has different meanings depending on the action that is performed.

The multitouch gestures associated with the movement of the main character are described in the section 4.2.1.1. The gestures associated with the interaction of the player with the game items, are described in the section 4.2.1.2. And the gestures associated with the attacks that can be performed by the player, are described in the section 4.2.1.3.

Figure 4.1: Game interface.

In the following sections these groups will be presented as well as the gestures that integrate each group. In the figures that illustrate the gestures, the black dot indicates the start point of each trace of the gesture.

4.2.1.1 Movement

This section describes the group of gestures, that are associated with movement actions of the player's character. The player can perform five different gestures that represent different movements of its character and these movements can be executed horizontally or vertically.

The player's character can only move horizontally in one or two positions. The gesture described in the figure 4.2, represents the gesture that the player has to perform to move the player's character to the next position on the left. The player only needs a finger to draw this gesture.

In the figure 4.3 is represented the gesture that the player has to execute, to move the player's character to the next position on the right. Such as the previous gesture, the player

Figure 4.2: Gesture to move the player's character to the left.

only needs a finger to perform it.

Figure 4.3: Gesture to move the player's character to the right.

The gesture described in the figure 4.4, represents the gesture that the player has to execute to move the player's character to the next two positions on the left. To perform this gesture the player needs to use two fingers to draw it.

Figure 4.4: Gesture to move the player's character two positions to the left.

The gesture described in the figure 4.5, represents a gesture that moves the player's character to the next two positions on the right. To execute this gesture the player needs to use two fingers to draw it.

Figure 4.5: Gesture to move the player's character two positions to the right.

The gesture described in the figure 4.6, represents the gesture that the player has to perform in order to execute a downward movement. This gesture can only be executed if the player's character is positioned on the top of a ladder.

Figure 4.6: Gesture that allows the player's character to perform a downward movement.

4.2.1.2 Interaction with items

This section describes the group of gestures associated with the interaction of the player's character with the different items in the game.

The player can perform six different gestures to interact with the environment of the game, but to execute the gestures the player's character has to be next to the element with whom it is going to interact. For example, if the player wants to open a chest, the player's character needs to be near the chest.

The story of the game puts the player in the role of a driller that needs to catch all the gems around the level in order to access the next level. This means that the player has to dig the ground to access to the lower platforms that contain the gems.

The gesture described in the figure 4.7, represents the drill action. When the player's character is positioned on top of a block of ground, if the player executes this gesture the player's character digs a hole and then falls into that hole. So the player must be very careful when executing this gesture because the player's character can be easily trapped.

Figure 4.7: Gesture to execute the action drill.

Throughout the levels there are scattered boulders that are blocking the path of the player. To push these boulders in order to clear the path, the player has to execute the gestures represented in the figure 4.8 or in the figure 4.9.

The gesture described in the figure 4.8, represents the action of pushing a boulder, one position to the right. This gesture will only have effect if the player's character is positioned next to the boulder.

Figure 4.8: Gesture to push an object to the right.

The gesture described in the figure 4.9, represents the action of pushing a boulder, one position to the left. This gesture will only have effect if the player's character is positioned next to the boulder.

Figure 4.9: Gesture to push an object to the left.

The main objective of the game is to catch all the gems from each level, but these gems are stored inside of chests. So first, the player has to reach the chest and open it. Only then, after opening the chest, the player is able to catch the gem.

The gesture described in the figure 4.10, represents the action of opening a chest. This gesture will only have effect when the player's character is near to a chest.

Figure 4.10: Gesture to open a chest.

The gesture describe in the figure 4.11, represents the action of catching a gem. This gesture will only have effect when the player's character is near to a gem.

Throughout the game, there are chests that are located in inaccessible regions. To access these places where the chests are, the player must interact with a mechanism, which

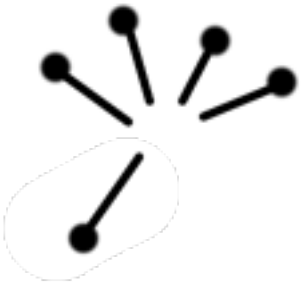

Figure 4.11: Gesture to catch a gem.

resembles a roller, that normally is located near to these places. When the player uses this mechanism, he begins to build a bridge connecting the player's character and the location of the chest.

To interact with this mechanism the player's character must be near the mechanism.

The gesture described in the figure 4.12, represents the gesture that the player has to execute to interact with the mechanism.

Figure 4.12: Gesture to interact with the roller.

4.2.1.3 Attack

This section describes the group of gestures associated with the attacks that can be performed by the player to destroy the enemies. The player can perform two different gestures that represent different types of attacks. In each level, there are several enemies, and the player has to destroy them to reach the chest with the gems.

The gesture described in figure 4.13, represents the gesture that the player has to execute to use the thunder attack. Using it, the player's character will throw a thunder to the enemies.

The gesture described in figure 4.14, represents the gesture that the player has to execute to use the rainbow attack. Using it, the player's character will throw a rainbow to the enemies.

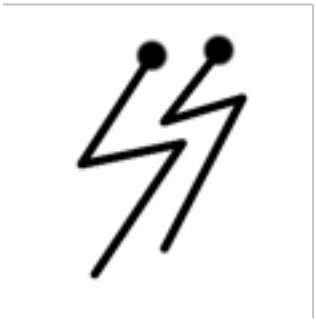

Figure 4.13: Gesture to use the thunder attack.

Figure 4.14: Gesture to use the rainbow attack.

4.2.2 Game Elements

The current section presents the description of the elements that integrate the levels of the game.

The elements are divided in groups, each group has a different meaning depending on what elements it contains. The scenario elements used to build the levels, are described in section 4.2.2.1. The interactive elements that the player has to interact with, to achieve the final objective, are described in the section 4.2.2.2. The game characters are described in section 4.2.2.3. The section 4.2.2.4 describes the different attacks that the player can execute.

4.2.2.1 Scenario Elements

The figure 4.15 presents the elements used to build the scenario. The elements *a)* and *d)* are blocks of ground, the player can use the action *drill* represented by the gesture in figure 4.7, but only if the player's character is positioned on top of a block of this type.

The element *b)* represents a stair. To interact with this element, the player must use the gesture represented in figure 4.6, to descend until reaching the ground.

The element *c)* represents a block of stone. This type of element does not support the action *drill*. It is used to build the game scenario or to block the path for strategic spots for

Figure 4.15: Elements used to build the game.

the player. To unblock the path, the player must use the push gestures represented in figure 4.8 or 4.9. These gestures shift the block a position to the right or left, depending on the used gesture.

4.2.2.2 Interactive Elements

The figure 4.16 presents the interactive elements of the game. The element *a)* represents a chest, the player has to open it to release the gem that is stored inside. To interact with this element, the player has to execute the gesture represented in figure 4.10.

The element *b)* represents a gem. The player has to catch all the gems around the level to complete the level, so this is the most important element in the game. As it was said before, the gems are always inside of a chest so the player must open first the chest to catch the gem inside. To catch the gem, the player has to execute the gesture represented in the figure 4.11.

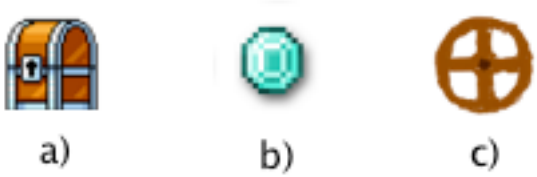

Figure 4.16: Interactive elements of the game.

The element *c)* represents an interactive mechanism. There are chests located in inaccessible places, because their is no physical path to reach these places. The player has to spin this mechanism to build a connection between the player's character and the chest. To interact with this element the player has to execute the roller gesture represented in figure 4.12.

4.2.2.3 Characters

The figure 4.17 presents the game characters. The character *a)* is the player's character, all the performed gestures are executed by this character.

The character *b)* and *c)* are the player's enemies. The character *b)* represents a head-bird and it moves faster than the character *c)* that represents a turtle. Both enemies do not have artificial intelligence, so they just move back and forth through the level.

Figure 4.17: Characters of the game.

4.2.2.4 Attacks

The figure 4.18 presents the attacks that the player can perform to destroy the enemies. The attack *a)* is a star-rainbow, to perform this attack the player has to execute the gesture represented in figure 4.14.

The attack *b)* is a thunder, to perform this attack the player has to execute the gesture represented in figure 4.13.

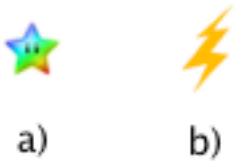

Figure 4.18: Attacks available in the game.

Each of these attacks can be performed anywhere in the game, to perform it the player only has to execute the corresponding gesture.

4.2.3 Game Interface

An example of a game interface is presented in figure 4.1. In this figure it is possible to identify most of the elements previously presented in the section 4.2.2.

To communicate with the game interface in order to interact with its character, the player must perform the gestures presented in the section 4.2.1.

Each gesture can be performed anywhere on the display, but must be executed one at a time because the gesture capture procedure only processes one gesture at a time.

If the gesture performed by the player is successfully executed then the action related with it, is performed and a counter that verifies if the gesture is well done, is incremented. In the figure 4.1, this counter is the number that is in front of the label *Gestures success*.

If the gesture performed by the player, is not well done then the counter that verifies that the gesture failed is decremented. In the figure 4.1 this counter is the number that is in front of the label *Gestures failed*.

Despite of the main objective of the player picking up all the gems around the world, the player must pay attention to the moves that needs to execute to reach the objective. One wrong movement and the player's character may be blocked and then the player has to restart the game.

Another thing to which the player must pay attention is the enemies around the level. The player must destroy all enemies and must be careful to never touch any of them or the player's character will die and the level restarts.

The victory condition of each level, is to catch all the gems and avoid being touched by the enemies.

4.3 Game implementation

The current section presents some details on the implementation of the game. In the section 4.3.1 the game structure is described, followed by the development environment of the game.

4.3.1 Game Structure

The game is divided in two main groups. The group *Framework* that contains the components *Gesture Capture*, *Recognizer* and *Templates*. And the group *Game* that contains the components *Game Interface* and the *Game Logic*. This division is depicted in the figure 4.19. Each component has a different function in the game.

- **Framework**
  - *Gesture Capture* is responsible for capturing the interactions between the player and the interface.
  - *Recognizer* is responsible for recognizing the meaning of the gesture executed by the player.

  - *Templates* is a set of gestures previously saved that are used in the recognition process.

- **Game**
  - *Game Interface* is responsible for managing the game interface and displaying the elements of the game.
  - *Game Logic* is responsible for managing the game logic, controlling the elements of the game and managing the resulting actions of the interactions between the player and the interface.

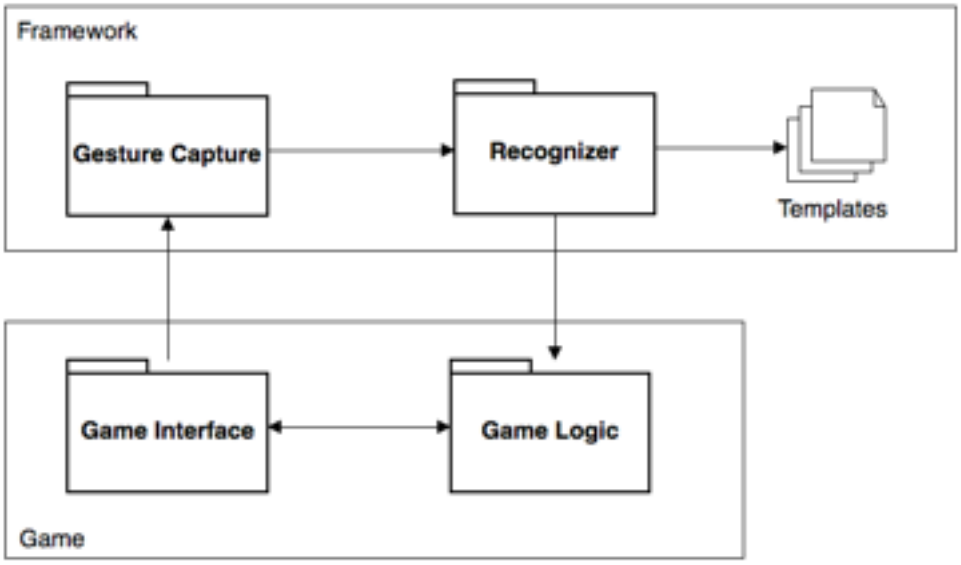

Figure 4.19: Game structure.

The *Game Interface* is the connection between the player and the game. The player, in order to interact with the game, has to send the input through the *Game Interface*. So whenever the *Game Interface* receives a new input of the player, it sends this input to the *Gesture Capture* because this component is responsible for assembling the gesture.

When the player finishes the gesture, the component *Gesture Capture* sends the data that identifies the gesture to the component *Recognizer*. In the *Recognizer* the gesture is compared with several *Templates* in order to identify which template is more similar to the executed gesture. In the end of the recognition process, the component *Recognizer* sends the result of the process to the game logic.

The component *Game Logic* needs to calculate if the gesture was successfully executed and then based on the gesture and on the positions of the game elements executes the next action of the game.

These results, will then be sent to the *Game Interface* that immediately processes the information and displays the results of the interaction between the player and the interface.

4.3.2 Development Environment

The game was designed to run on the iPad, a mobile device of Apple under the iOS operative system. It was developed on a MacBook Pro 1,2 running Mac OS X 10.6.7 (Snow Leopard). The IDE used was Xcode and the programming language used was Objective-C.

In order to accelerate the development of the game, it was used an open-source 2D game framework, *Cocos2D*. This framework is meant for building 2D games or other graphical/interactive applications. It is easy to use, fast, flexible and free.

Chapter 5

Tests and Results

This chapter describes a survey performed by 15 participants, with the objective of evaluating the effects of the inclusion of multitouch gestures in a game interface.

During the tests, each participant had to play the game, described in the chapter 4. And in the end of each test, the participant had to fill out a questionnaire about their experience with the game interface.

This chapter also describes the procedure followed by the participants during the tests, as well as their attributes and the final results of the survey. In the end, it is presented an analysis of the results of the survey, with an evaluation of the usability of the game interface.

5.1 Usability Concepts

The best way to understand the effects of the incorporation of multitouch gestures in a game interface, it is to measure its usability.

Usability is a measure of the quality of the user experience when interacting with an application or interface. Normally, usability refers to how well users can learn and use an interface to achieve their goals and how satisfied they are with that process [BMO08]. An usable interface is more efficient and satisfying to use and easier to learn.

It is important to understand that usability is not a single property of a user interface. It is composed by the following features [Nie93]:

- Ease of learning: How fast can a user, who never seen the user interface before learn it sufficiently well, to accomplish basic tasks?

- Efficiency of use: Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?

- Memorability: If a user has used the system before, can he or she remember enough to use it effectively the next time or does the user have to start over again learning everything?
- Errors: How often do users make errors while using the system, how serious are these errors, and how do users recover from these errors?
- Subjective satisfaction: How much does the user like using the system?

5.2 Procedure

A pilot study, with one participant, was previously conducted. The objective of this study was to practice the test procedure and to determine how long the tests would take, and to point out unforeseen bugs in the game.

In the end of this pilot study, it was decided that each test should take at least 20 minutes. The procedure followed in each test session is described in the next steps:

1. The participants were asked to fill out a questionnaire to evaluate their computer literacy, game experience, and touch experience. The questionnaire is presented in the appendix A.
2. The participants were presented with a small introduction about the thesis subject and the framework for gestures recognition, described in the chapter 3.
3. The participants were introduced to the game. It was explained the game objectives and how to interact with the game interface.
4. The participants were presented with a practice session, where they had to play the tutorial levels to gain experience with the game interface.
5. The participants were asked to complete a level of the game without any help. If any participant required external assistance to perform some gesture, it was provided a reference card with the gesture.
6. The participants were asked to fill out a questionnaire, presented in appendix B, about their usage experience of the game interface.

5.3 Participants

The participants were recruited in the company where the thesis was conducted and in the computer labs of the faculty. Their ages were comprised between 23 and 33 years, and the number of male participants was 14 and only one female participant.

In the table 5.1 is represented the distribution of the participants by their area of competence.

Competence Area	Number of Participants
Computer Science	12
Art and Design	3

Table 5.1: Distribution of participants based in its competence area.

An important feature to measure in the participants is, how comfortable they feel using a multitouch device. It is important to understand the different backgrounds of the participants, because these differences could have impact in the results of the survey. The table 5.2 represents the distribution of the participants by their familiarity with multitouch technology.

Multitouch Familiarity	Number of Participants
Very Bad	0
Bad	0
Normal	4
Good	4
Very Good	7

Table 5.2: Distribution of participants based in its familiarity with the multitouch technology.

5.4 Questionnaire details

The current section presents with more detail the survey, the questionnaire used to measure the usage experience of the game interface is in the appendix B. The section 5.4.1 describes the type of scale used in the questionnaire. The section 5.4.2 describes the statements of the questionnaire, the objective of each statement and their results. In the section 5.5 is presented an analysis to the results of the survey.

5.4.1 Scale

The type of scale used in the questionnaire was the *Likert-scale*, a psychometric scale commonly used in questionnaires. When the participants respond to a *Likert* questionnaire item, they specify their level of agreement or disagreement on a symmetric agree-disagree scale for a series of statements and the scale captures the intensity of their feelings [Bur08].

The format chosen for the *Likert-scale* was the four-level item. Therefore, each statement should be answered using the following options:

1. Strongly Disagree

2. Disagree
3. Agree
4. Strongly Agree

Likert is a bipolar scaling method, measuring either positive or negative response to a statement. It is not normal to have a four-point scale, in this case it was chosen to avoid the *"neither agree nor disagree"* choice, preventing invalid answers.

5.4.2 Statements

In the end of each test, the participant had to fill out a questionnaire, described in the appendix B, reporting their usage experience of the game interface. The statistics of each statement are available in the appendix C.

The questionnaire was composed by a set of statements, with the objective of measuring different features in the usability of the game interface. The following list presents - the statements and their objectives.

1. *"It is easy to use"* has the objective to understand how fast can a user who never seen the game interface, learn it sufficiently well to accomplish basic tasks in the game.
2. *"It is fun to use"* has the goal to perceive how the participants react to the interaction with the game interface. So it was important to know, if they thought that the game interface was boring and not challenging or by the contrary.
3. *"It is useful"* was important to understand how far the use of these interfaces enhance the interactions between the player and the game.
4. *"I learned to use it quickly"*, through this statement it was possible to understand, once the user has learned to use the game interface, how fast he can accomplish basic tasks in the game.
5. *"Using it is effortless"* was important to understand how effortless is to use these type of game interface, specially when the interface supports a variety of multitouch gestures.
6. *"I easily remember how use it"* was useful to understand, if an user that had experience with the game interface, will remember enough to use it effectively another time or does he needs to learn everything again.
7. *"I quickly could become skillful with it"* has the objective to evaluate the degree of confidence related with the skills acquired during the use of the game interface.

8. *"It is user friendly"* was important to evaluate if it is user friendly using multitouch gestures in mobile game context.

9. *"It would make things I want to accomplish easier to get done"*, through this statement, was possible to understand how far the multitouch gestures can replace the common ways of interaction.

10. *"Both occasional and regular players would like it"* had the objective to evaluate the participants' opinion about the level of acceptance of these interfaces by others players.

5.5 Results and Analysis

The survey was conducted by 15 participants, in the end of each practice test, the participants had to fill out a questionnaire, composed by 10 statements, reporting their usage experience with the game interface. Before presenting the analysis of the results, it is presented some decisions in the game design and implementation that could have had influence on the results of the survey. And after it, a deeper analysis of the results of the survey will be performed based on the features that define usability.

There are two important decisions that could have had influence on the results of the survey, the selection of the game type and the choice of the multitouch gestures to integrate the game.

The selected game type was a two-dimension adventure game. And it was chosen because the most common way to interact with its interface, it is by using a game controller or a keypad. Therefore, creating an usable interface, for this game type, based on multitouch gestures was an interesting challenge.

The choice of multitouch gestures was based on the perception of the actions that the player had to perform. For example, to catch a gem the player needs to perform the gesture in the figure 4.11, this gesture requires that the player uses 5 fingers. But in the real life, a person only uses 5 fingers to catch something big, if not, it is more common to use 2 or 3 fingers. So the multitouch gestures could have a different mental representation, depending the person that performs it.

The following sections presents a deeper analysis of the survey based on the features, described in the section 5.1.

5.5.1 Ease of learning

The chart shown in the figure 5.1, presents the results of the attribute *Ease of learning* and was based in the following statement *"I learned to use it quickly"*. It is possible to identify that 93% of the answers were positive feedback.

The measure of this attribute was important to understand how easy can a user, without any background experience with the game interface, perform the basic tasks. And the results were very clear: based on the positive feedback of the participants, it is possible to say that most of the participants could perform easily the basic actions with only a few minutes of usage of the game interface.

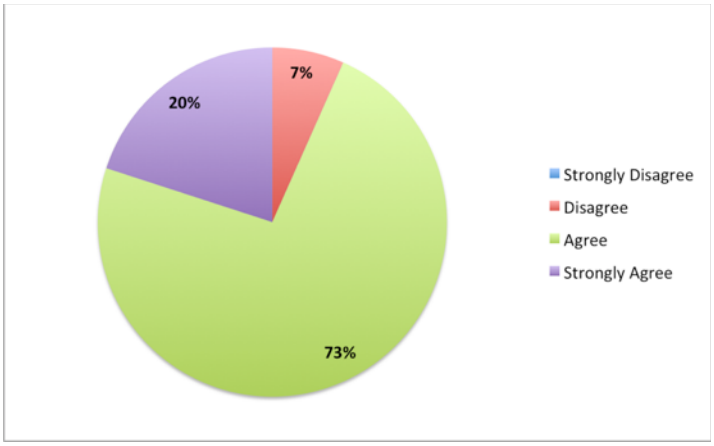

Figure 5.1: Results of the attribute *Ease of learning*.

But during the tests, it was also possible to observe that the participants learned more easily gestures related with movement and interaction. And by the contrary the attack gestures were the most difficult to learn.

This trend can be explained with the analysis of the form of the gestures. It was not easy to map into a gesture the meaning of a thunder or rainbow attack. The concept followed to design these gestures were to draw the first image that comes to the mind when thinking in the attack. For example, if a participant thought to use a thunder attack, he had to draw a thunder. But during the tests, the most participants had different interpretations of the meaning of a thunder and this misunderstanding had a direct impact in the way that they represented the attack.

To improve the learning process of the participants in these specific type of gestures, it was extended the training period. Therefore, expending more time in the practice sessions the success rate improved.

5.5.2 Efficiency of use

The chart showed in the figure 5.2, presents the results of the attribute *Efficiency of use* and was based in the following statements - *"It is easy to use"*; *"Using it effortless"*; *"It is user friendly"* and *"It would make the things I want to accomplish easier to get done"*.

Analyzing the chart, the positive feedback was 82% and the negative feedback was 18%. Although the value of negative feedback is not a significantly high value it is important

to understand what it means. During the test there were some complains related with the variety and complexity of the selected gestures.

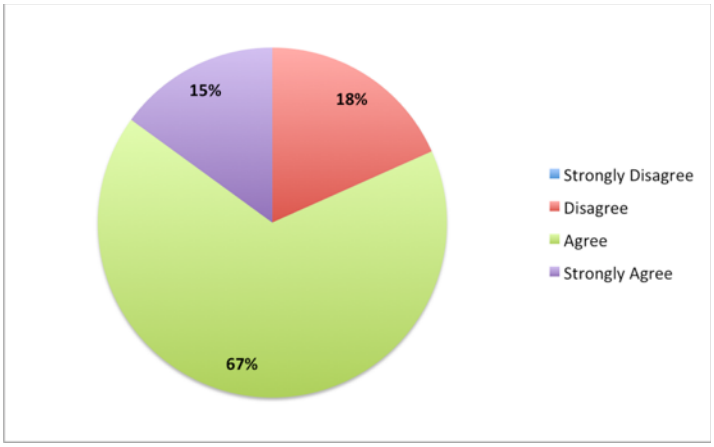

Figure 5.2: Results of the attribute *Efficiency of use*.

Some participants failed the execution of certain gestures because they had forgotten the number of fingers to use in those specific gestures. The participants knew the form of the gesture but the problem was, how many fingers they had to apply in order to perform the gesture.

A good solution for this problem, it was to conduct a study with real users in order to evaluate the most suitable gesture for each action. This study would be useful to understand, the user's perception of the different actions. Using this knowledge would be possible to have a better definition of each gesture. But due to time constraints, it was impossible to realize this study.

5.5.3 Memorability and Errors

The attributes *Memorability* and *Errors* are connected, whenever a participant had problems to remember how to perform a specific gesture, the most common form he used to surpass it, was to execute the gesture in a wrong way, introducing errors in the game.

The chart showed in the figure 5.3, presents the results obtained by the attributes *Memorability and errors*. The results were based in the following statements - *"I easily remember how to use it"* and *"I quickly could become skillful with it"*.

The measure of these attributes was important to understand if the users had problems to remember which gestures to perform to accomplish a specific task.

Although the excellent results obtained in the survey, forgetting how many fingers to apply to perform the gesture or even mixing the gestures was common in the participants. But these problems are normal, because the participants only had 5-10 minutes to learn 13 different multitouch gestures.

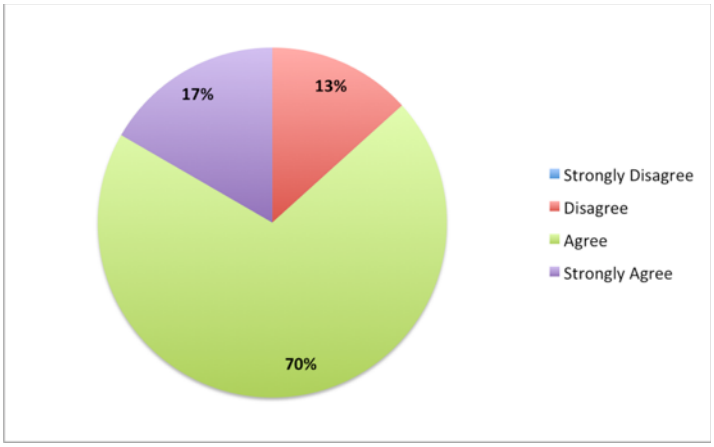

Figure 5.3: Results of the attribute *Memorability and Errors*.

A possible solution for this problem was already refereed in the previous feature. But there were other improvements that could have helped the lack of memory.

Another solution was to use the same number of fingers to perform similar actions. For example, to perform the movement actions the participants had to use gestures with 2 fingers, to perform the attack actions the participants had to use gestures with 3 fingers and to perform interaction actions the participants had to use gestures with 4 fingers.

5.5.4 Subjective satisfaction

In the figure 5.4 is presented a chart that describes the results obtained by the attribute *Subjective satisfaction* and were based on the following statements - *"It is fun to use"*; *"It is useful"* and *"Both occasional and regular users would like it"*.

Measuring the satisfaction of usage of the game interface, was probably one of the most important aspects of this survey. It reflects how much the user enjoyed the experience of the interaction by using multitouch gestures as main game interface.

And after observing the results of the survey, this attribute had the highest value on the option *Strongly Agree* and one of the lowest values on the negative feedback. Which means that the participants really liked using the multitouch gestures for the interactions.

Other way to evaluate the satisfaction of the participants would be, to implement the same game but with conventional forms of interaction. And then, compare the degree of satisfaction of the participants using both interfaces. But due to time constraints, it was impossible to implement this functionality.

5.5.5 Conclusions

The chart shown in the figure 5.5, presents the global results of the survey. It is possible to identify in the chart that 67% of the answers of the participants were agreeable and 20% strongly agreeable with the statements, which constitutes 87% of positive feedback against

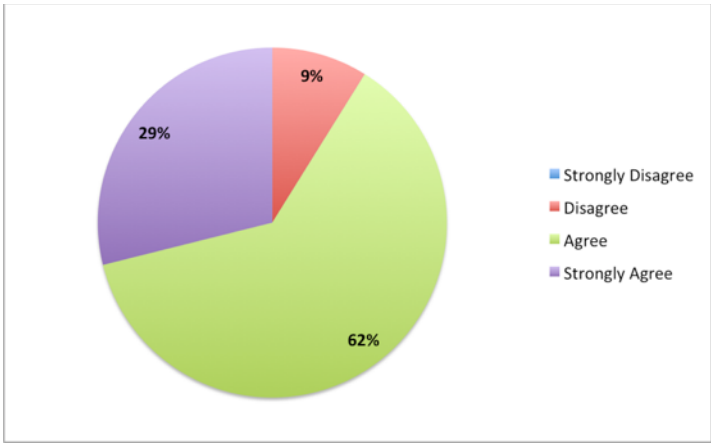

Figure 5.4: Results of the attribute *Subjective Satisfaction*.

only 13% of negative feedback. The values of the chart were based in a universe of 150 answers.

So it is possible to conclude that the integration of multitouch gestures in a gaming environment for mobile devices is usable and enhances the communication between the player and the game interface.

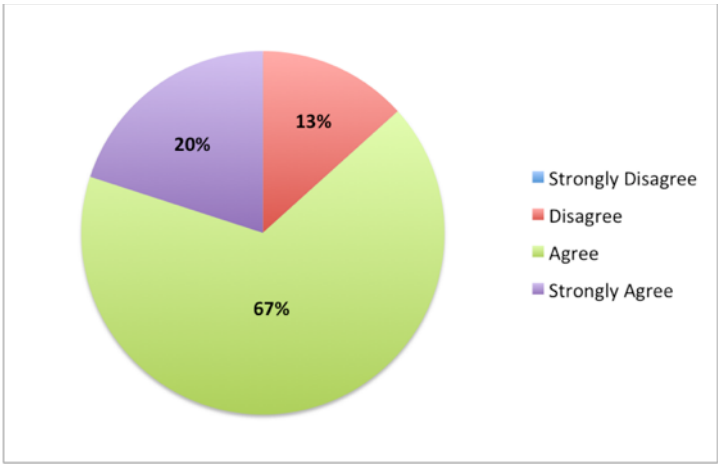

Figure 5.5: Results overall.

Chapter 6

Conclusion and Future Work

This document, developed in the aim of a Dissertation of the Master in Informatics and Computing Engineering, describes the research conducted in the area of the recognition of multitouch gestures:

- The project was motivated by the possibility of exploring new interactions using multitouch gestures as a main interface between users and applications, and applying the knowledge obtained, in a interactive application for a mobile device.

- Before the beginning of the project, an investigation was conducted in order to understand which were the most important techniques used to recognize multitouch gestures. The study allowed to conclude that this is an interesting area of investigation, mainly for the new forms of interaction that can emerge of these types of recognizers. Since the project was to be implemented on a resource-constrainted device, the better technique to use was the algorithm *$1 Recognizer*, described in article [WWL07].

- The first step of the development process, was to implement a prototype of the algorithm *$1 Recognizer* to recognize multitouch gestures. The objective of the prototype was to evaluate the performance of the solution developed.

- Despite its good performance, the solution developed still had some limitations, mainly in the preprocessing process that was applied to the input gesture, the limitations are described in more detail in the section 3.1.4. But most of these limitations could be corrected with the implementation of the improvements described in the section 3.1.5, but due to time constraints it was impossible to implement them.

- With the algorithm for the recognition developed, the following step was to implement the algorithm in the deployment environment. The mobile device used for the deployment was an iPad. It was built a framework, that helped in the prototyping process of the multitouch gestures. The framework is described in the section 3.2.

- The framework was used in the prototyping process of the gestures which would later be used in the game interface. During the prototyping process, a small survey was conducted in order to evaluate which were the most interesting gestures to use in a gaming environment.
- The next step of the development process, was to create a game with multitouch gestures as a main interface. The game type selected was, a two-dimensional adventure game, presented in the chapter 4.
- To evaluate the usability of the game interface, a survey with real participants was conducted. Each participant followed the procedure described in the section 5.2. And in the end of each test, the participants had to fill out the questionnaire described in the appendix B.

Some artifacts were not shown on the main body of the dissertation but were included as appendix. Appendix A shows the background questionnaire that the participants of the tests had to fill out. Appendix B presents the questionnaire about the usage experience of the game interface, that the participants had to fill out in the end of the survey. Finally, the Appendix C presents the statistics of the results of the survey.

6.1 Goal Achievement

The work carried out enabled the presentation of an algorithm for the recognition of multitouch gestures. This algorithm was embedded in a framework, that allows the users to prototype different multitouch gestures and use them in other applications.

An example of this incorporation of the framework in a interactive application was the game implemented in the chapter 4. The game interface uses only multitouch gestures to interact with the players.

In order to evaluate the usability of the game interface, a survey with real participants was conducted. The results of the survey were very clear, the game interface is usable. The results of the survey are presented in the appendix C.

6.2 Future Work

The work done until this moment enables to identify the following set of tasks to be made in the future, which could potentially improve the quality of the research carried out:

- The initial planning of the project herein described proved to be undervalued and time was not enough to meet all the initial proposed objectives. In fact, one goal that was soon discarded, due to time restrictions was the use of more input sensors,

like the accelerometer and gyroscope. It would be of great interest to measure the usability of a multimodal interface, instead of using just the multitouch technology.

- To enhance the results of the discussion of the survey, having a larger group of participants would be essential. And instead of just testing the multitouch interface, it would be interesting to implement the same game but with a common interface and making a benchmark between the two interfaces in order to evaluate which was the most usable.

- In what concerns to the algorithm used in the recognition process, there were some improvements to be implemented that would had a direct impact in the success rate of the recognition process, such as the implementation of a new sampling process of the gesture, this process is described in the section 3.1.5. And the implementation of a closed-form solution to find a rotation that leads to the minimum angular distance between the gesture and the template, this improvement is described in the section 3.1.5.

- The source code of the framework used in the recognition process lacks of a properly designed software architecture, one that could be easily expanded to be incorporated in other games.

- In what concerns with the multitouch gestures used in the game, there are some improvements that would enhance the the playability of the game. These improvements are:

    1. During the selection of what gestures should be integrated in the game, it should have been conducted a study with real participants, in order to evaluate the most suitable gestures for the different actions of the game. This study would be useful to understand the participants' perception of each action, and using this knowledge would be possible to have a better definition of each gesture.
    2. During the tests, there were some complains related with the variety and complexity of the selected gestures. The participants knew the form of the gesture, but the problem was, to remember how many fingers they had to apply in order to perform the gesture. A good solution, was to use the same number of fingers to perform similar actions in the game.

- Finally, to understand which were the most suitable games for the incorporation of multitouch gestures, it should have been performed a previous study with real participants with a high background in multitouch game experience.

References

[Ada09] Ernest Adams. *Fundamentals of Game Design*. Berkeley, New Riders, 2009.

[BIB+11] Regina Bernhaupt, Katherine Isbister, John Buchanan, Daniel Cook, and Dave Warfield. Games and hci: perspectives on intersections and opportunities. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 351–354, New York, NY, USA, 2011. ACM.

[BMO08] Regina Bernhaupt, K Mihalic, and Marianna Obrist. *Methods for Usability Evaluation of Mobile Devices*. ACM, 2008.

[Bur08] Alvin Burns. *Basic marketing research : using Microsoft Excel data analysis*. Pearson Prentice Hall, Upper Saddle River, N.J, 2008.

[CPG07] Qing Chen, Emil M Petriu, and Nicolas D Georganas. 3d hand tracking and motion analysis with a combination approach of statistical and syntactic analysis. *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, pages 56–61, 2007.

[Day90] J E Dayhoff. *Neural Network Architectures: An Introduction*. Van Nostrand Reinhold, 1990.

[DET90] R F Dillon, Jeff D Edey, and Jo W Tombaugh. Measuring the true cost of command selection: techniques and results. *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people CHI 90*, (April):19–26, 1990.

[DO96] R Drossu and Z Obradovic. Rapid design of neural networks for time series prediction. *IEEE Computational Science and Engineering*, 3:78–89, 1996.

[DP73] David H DOUGLAS and Thomas K PEUCKER. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[Eco10] Growth in mobile applications: Apps and downs, 2010.

[FWSB07] C Forlines, D Wigdor, C Shen, and R Balakrishnan. Direct-touch vs. mouse input for tabletop displays. *Proceedings of the SIGCHI conference on Human factors in computing systems CHI 07*, page 647, 2007.

[Gro09] NUI Group. Nui group community faqs, July 2009.

[Hea08] Jeff Heaton. *Introduction to Neural Networks for Java*. Heaton Research, 2008.

[HP95] Thomas S Huang and Vladimir Pavlovic. Hand gesture modeling, analysis, and synthesis. *Proceedings*, 1995.

[JDM00] A K Jain, Robert P W Duin, and J Mao. Statistical pattern recognition : A review. *Analysis*, 22(1):4–37, 2000.

[Kad02] Wan Kadous. *Computer Based Machine Learning Unit*. PhD thesis, University of New South Wales, 2002.

[KAD09] Kenrick Kin, M Agrawala, and T DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. *Text*, pages 119–124, 2009.

[KS04] L B Kara and T F Stahovich. An image-based trainable symbol recognizer for sketch-based interfaces. *AAAI Fall Symposium Series 2004 Making PenBased Interaction Intelligent and Natural*, page 99–105, 2004.

[Li10] Yang Li. *Protractor: a fast and accurate gesture recognizer*, pages 2169–2172. ACM, 2010.

[Liu10] Weiyuan Liu. Natural user interface- next mainstream product user interface. *International Journal*, pages 203–205, 2010.

[MH06] Tomer Moscovich and John F Hughes. *Multi-finger cursor techniques*, page 1–7. Canadian Information Processing Society, 2006.

[MH08] T Moscovich and John F Hughes. Indirect mappings of multi-touch input using one and two hands. *Proceeding of the twentysixth annual CHI conference on Human factors in computing systems CHI 08*, page 1275, 2008.

[ML08] Dave Mark and Jeff LaMarche. *Beginning iPhone development : exploring the iPhone SDK*, volume 7. Apress, 2008.

[Mos07] T Moscovich. Principles and applications of multi-touch interaction. *Brown University*, 54(May 2007):1–114, 2007.

[Nie93] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[Nie08] Gerrit Niezen. *The Optimization of Gesture Recognition Techniques for Resource-constrained Devices*. PhD thesis, University of Pretoria, 2008.

[NK08] B B Nasution and A I Khan. A hierarchical graph neuron scheme for real-time pattern recognition. *IEEE Transactions on Neural Networks*, 19(2):212–229, 2008.

[NPM+03] Cambridge New, York Port, Chester Melbourne, William T Vetterling, Saul A Teukolsky, William H Press, and Brian P Flannery. *Numerical Recipes in C*, volume 2nd. Cambridge University Press, 2003.

[Rai99] Roope Raisamo. *Multimodal Human-Computer Interaction: a constructive and empirical study*. PhD thesis, University of Tampere, 1999.

[RJ86] Lawrence R Rabiner and B H Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[SC78] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, 26(1):43–49, 1978.

[SD10] Steven C Seow and Ph D. Natural user interfaces : The prospect and challenge of touch and gestural computing. *Human Factors*, pages 4453–4455, 2010.

[Wat93] Richard Watson. A survey of gesture recognition techniques technical report tcd-cs-93-11. *Language*, (July):1–31, 1993.

[WM10] Daniel Wigdor and Gerald Morrison. Designing user interfaces for multi-touch and surface-gesture devices. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 3193–3196, New York, NY, USA, 2010. ACM.

[WMW09] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. User-defined gestures for surface computing. *Proceedings of the 27th international conference on Human factors in computing systems CHI 09*, page 1083, 2009.

[WWL07] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. Gestures without libraries, toolkits or training: a 1 recognizer for user interface prototypes. *Proceedings of the 20th annual ACM symposium on User interface software and technology UIST 07*, pp:159, 2007.

REFERENCES

Appendix A

Background Experience Questionnaire

Questionnaire:

Rate your level of computer literacy.
Very Bad - Bad - Normal - Good - Very Good

How often do you play games?
Never - Rarely - Normal - Regularly - Daily

How often do you play adventure games?
Never - Rarely - Normal - Regularly - Daily

Do you own a multitouch device (e.g. iPhone, iPod Touch, iPad, Android, etc)?
Yes - No

How confortable do you feel using a multitouch device?
Very Bad - Bad - Normal - Good - Very Good

Appendix B

Game Experience Questionnaires

Questionnaire:

It is easy to use.
Strongly Disagree - Disagree - Agree - Strongly Agree

It is fun to use.
Strongly Disagree - Disagree - Agree - Strongly Agree

It is useful.
Strongly Disagree - Disagree - Agree - Strongly Agree

I learned to use it quickly.
Strongly Disagree - Disagree - Agree - Strongly Agree

Using it is effortless.
Strongly Disagree - Disagree - Agree - Strongly Agree

I easily remember how to use it.
Strongly Disagree - Disagree - Agree - Strongly Agree

I quickly could become skillful with it.
Strongly Disagree - Disagree - Agree - Strongly Agree

It is user friendly.
Strongly Disagree - Disagree - Agree - Strongly Agree

It would make the things I want to accomplish easier to get done.
Strongly Disagree - Disagree - Agree - Strongly Agree

Both occasional and regular users would like it.
Strongly Disagree - Disagree - Agree - Strongly Agree

Appendix C

Questionnaire Results and Statistics

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	1	7%
Agree	10	67%
Strongly Agree	4	26%

Table C.1: Results of the statement *"It is easy to use"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	0	0%
Agree	9	60%
Strongly Agree	6	40%

Table C.2: Results of the statement *"It is fun to use"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	0	0%
Agree	12	80%
Strongly Agree	3	20%

Table C.3: Results of the statement *"It is useful"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	1	7%
Agree	11	74%
Strongly Agree	3	19%

Table C.4: Results of the statement *"I learned to use it quickly"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	4	27%
Agree	9	60%
Strongly Agree	2	13%

Table C.5: Results of the statement *"Using it is effortless"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	4	27%
Agree	9	60%
Strongly Agree	2	13%

Table C.6: Results of the statement *"I easily remember how to use it"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	0	0%
Agree	12	80%
Strongly Agree	3	20%

Table C.7: Results of the statement *"I quickly could become skillful with it"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	2	13%
Agree	11	74%
Strongly Agree	2	13%

Table C.8: Results of the statement *"It is user friendly"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	4	27%
Agree	10	66%
Strongly Agree	1	7%

Table C.9: Results of the statement *"It would make things I want to accomplish easier to get done"*.

Value	Frequency	Percentage
Strongly Disagree	0	0%
Disagree	4	27%
Agree	7	46%
Strongly Agree	4	27%

Table C.10: Results of the statement *"Both occasional and regular users would like it"*.